

TD 10: XSLT

Vé

```

<body>
  <table border=1 cellpadding=2 cellspacing=0>
    <tr>
      <th>Titre</th>
      <th>Auteurs</th>
    </tr>
    <tr>
      <td>L'essentiel de XML</td>
      <td>Olivier Carton</td>
    </tr>
    <tr>
      <td>War and Peace</td>
      <td>Leon Tolstoï</td>
    </tr>
  </table>
</body>
</html>

```

Titre	Auteur
L'essentiel de XML	Olivier Carton
War and Peace	Leon Tolstoï

Voir un [aperçu du résultat demandé](#). Corrigés: [sortie HTML](#), [XSL](#)

2. Essayez-le avec le fichier exemple!

- `xsltproc votre.xsl 1.xml > 1.html`
- Ouvrez `1.html` dans votre navigateur et vérifiez qu'il s'affiche correctement
- Corrigez votre `.xsl` si besoin et recommencez

3. Modifiez votre XSLT pour que, à l'intérieur des titres, le titre soit en gras (HTML: ``), et le contenu des éléments `<subtitle>` soit affiché après un saut de ligne (en HTML: `
`) et en vert foncé (``).

Voir un [aperçu du résultat demandé](#). Corrigés: [sortie HTML](#), [XSL](#)

4. Modifiez votre XSLT pour que les auteurs multiples soit séparés par des slash, comme ceci: `Olivier Carton / Leon Tolstoï`.

Indices / Conseils:

- Regardez le fichier `1.xml` !
- Essayez d'éviter `<xsl:for-each>`. On préfère toujours `<xsl:apply-templates>`.
- Supportez également les `<bibtex:author>` qui ne contiennent que du texte sans `<bibtex:person>`
- Attention: les `<bibtex:person>` peut contenir des sous-éléments; dont on voudra seulement extraire le texte.
- [Facultatif] Faites en sorte que les slash n'apparaissent pas avant le premier auteur, ni après le dernier!

Voir un [aperçu du résultat demandé](#). Corrigés: [sortie HTML](#), [XSL](#)

5. Ajoutez une colonne pour l'année de chaque entrée; et triez les entrées par année, grâce à `<xsl-sort>`.

Voir un [aperçu du résultat demandé](#). Corrigés: [sortie HTML](#), [XSL](#)

6. Ajoutez une colonne qui contiendra la position *originale* de chaque entrée dans le fichier source (#1, #2, ...).

Voir un [aperçu du résultat demandé](#). Corrigés: [sortie HTML](#), [XSL](#)

7. (*) Modifiez l'expression précédente pour que la position soit indexée à partir de #0, et non de #1.

Voir un [aperçu du résultat demandé](#). Corrigés: [sortie HTML](#), [XSL](#)

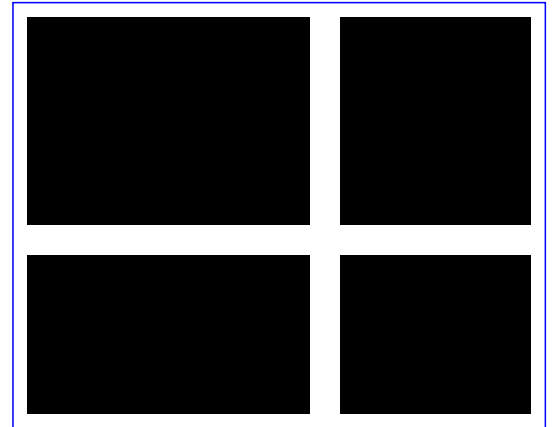
8. Ajoutez une colonne donnant le type de l'entrée, sans le préfixe `bibtex:` (eg. `book`, `article`, ...). Colorisez les **cases** de cette colonne: les `book` et `article` en rouge et bleu (respectivement); les autres inchangés. On préférera pour cela modifier l'élément `<td>` de la sortie en lui ajoutant un attribut `style`, comme expliqué [ici](#).

Exercice 2: SVG

On travaille sur [un fichier SVG](#) (à gauche, son contenu; à droite son rendu; tel que vous le voyez en ouvrant le fichier `.svg` dans un navigateur Web):

```
<?xml version="1.0" standalone="no"?>
<svg width="700" height="560" version="1.1"
    xmlns="http://www.w3.org/2000/svg">
  <desc>Four separate rectangles
</desc>
  <rect x="20" y="20" width="370" height="272"/>
  <rect x="430" y="20" width="250" height="272"/>
  <rect x="20" y="332" width="370" height="208"/>
  <rect x="430" y="332" width="250" height="208"/>

  <!-- Show outline of canvas using 'rect' element -->
  <rect x="1" y="1" width="698" height="558"
    fill="none" stroke="blue" stroke-width="2" />
</svg>
```



1. Écrire une stylesheet XSLT qui, appliquée au fichier `2.svg`, donne le nombre de rectangles.

Aides:

- Quand le document source déclare une namespace par défaut (c'est le cas ici!); il est impératif de re-déclarer cette namespace dans la feuille de style XSLT, en lui **donnant un préfixe** non vide. Vos expressions XPath devront ensuite utiliser ce préfixe pour référencer des éléments. [Explication sur stackoverflow](#)
- Utilisez `<xsl:text>
</xsl:text>` pour ajouter un saut de ligne.

Exemple de sortie souhaitée: `Nombre de rectangles: 5`

[Corrigé](#)

2. Modifiez votre XSLT pour afficher également les aires des rectangles. On supposera que toutes les unités du SVG sont en centimètres.

Exemple de sortie souhaitée:

```
Nombre de rectangles: 5
Aires des rectangles:
100640 pixels
68000 pixels
76960 pixels
52000 pixels
389484 pixels
```

[Corrigé](#)

3. Modifiez votre XSLT pour qu'il ne prenne en compte que les rectangles **remplis** (regarder l'attribut `fill`), et qu'il calcule les aires en **pourcentage** de la surface totale.

Exemple de sortie souhaitée:

```
Nombre de rectangles: 4
Aires des rectangles:
25.6734693877551 %
17.3469387755102 %
19.6326530612245 %
13.265306122449 %
```

[Corrigé](#)

4. Créez un XSLT qui transforme un SVG source en le même SVG, réduit de 50%.

Aides:

- N'oubliez pas d'ajuster `xsl:output`

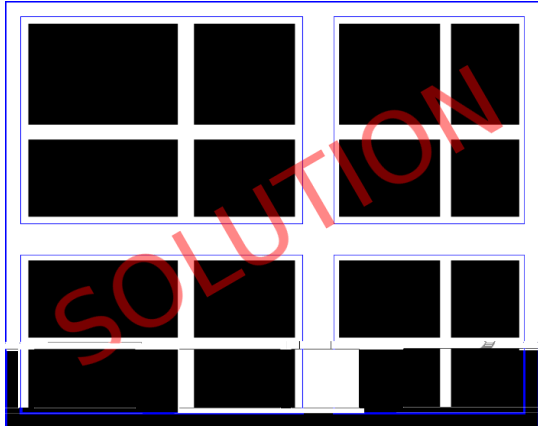
- Si on éditait directement le fichier SVG, la solution la plus simple serait d'insérer un élément `<g>` juste en dessous de l'élément racine `<svg>` avec un attribut `transform` adéquat (cf [SVG reference](#))
- Le XSLT peut utiliser `<xsl:copy>` pour recopier l'élément courant (sans ses attributs ni contenu), et `<xsl:copy-of>` pour recopier des parties d'un élément, telles que ses attributs, et/ou son contenu.

Appliquez ce XSLT au fichier 2.svg: `xsltproc votre.xsl 2.svg > 2.4.svg`, et ouvrez la sortie 2.4.svg dans votre navigateur.

Réparez votre XSLT jusqu'à ce que ça marche!

[Corrigé](#)

- (*) Créer un XSLT qui transforme 2.svg en remplaçant chaque rectangle **rempli** par une copie *réduite* du document SVG originel (correspondant à la taille du rectangle remplacé).
Ça devrait ressembler à l'image ci-dessous:



[Corrigé](#)

- Modifiez, si besoin, le XSLT pour qu'il puisse s'appliquer récursivement.
Appliquez-le 3 fois:

```
xsltproc votre.xsl 2.svg > 2.6.svg
xsltproc votre.xsl 2.6.svg > 2.66.svg
xsltproc votre.xsl 2.66.svg > 2.666.svg
```

Et vérifiez qu'il ressemble à ça:



[Corrigé](#)

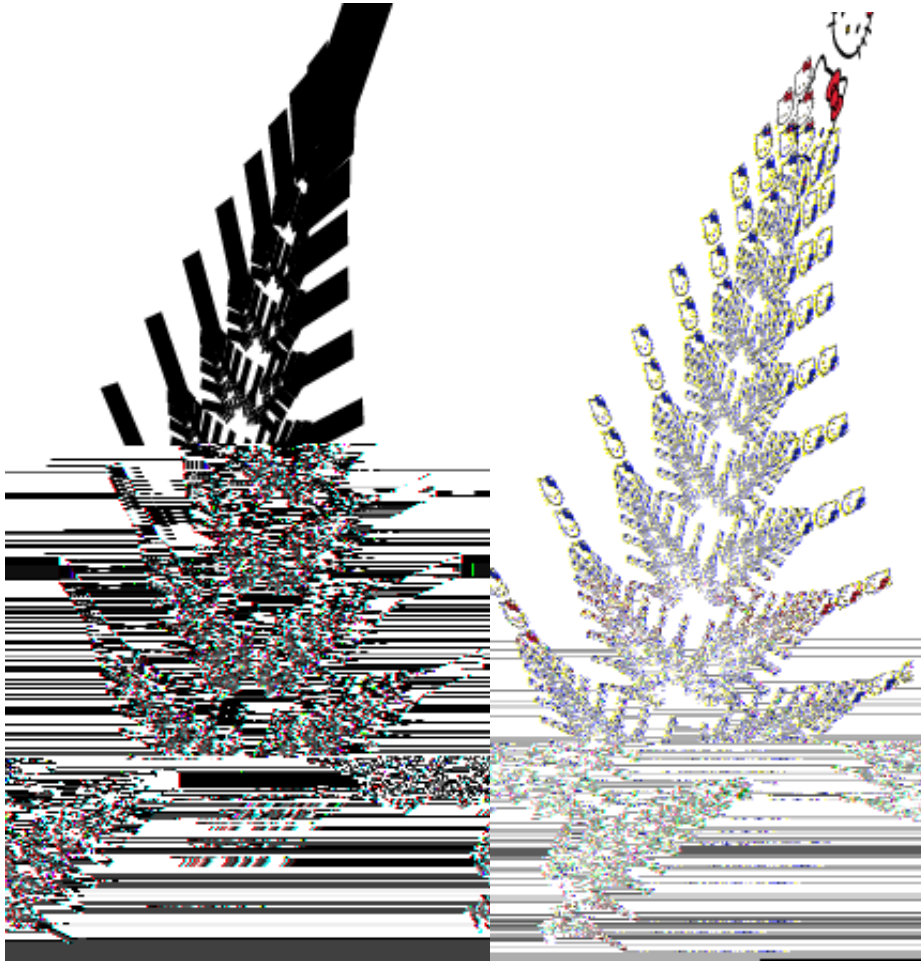
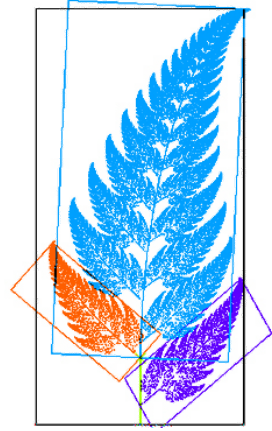
- (**) En s'aidant du graphique ci-contre (cliquer pour l'agrandir), créer un XSLT qui permet, en l'appliquant récursivement, de transformer toute image SVG de proportions 200x450 en [fougère de Barnsley](#).

Pour ajuster vos transformations, utilisez le fichier [2.7.svg](#): `xsltproc votre.xsl 2.7.svg > test.svg`. Une fois ajusté; appliquez-le 10 fois de suite sur [2.7.0.svg](#) à l'aide de cette commande (copiez-collez tout):

```
XSL=votre.xsl
for i in {0..10}; do
  echo "Iteration #$i"
```

```
(( j=i+1 ))
xsltproc $XSL 2.7.$i.svg > 2.7.$j.svg
j=$i
done
```

Vous pouvez également essayer avec d'autres images 200x450, comme [Hello Kitty](#) (en le sauvegardant sous `2.7.0.svg`, le script marchera sans modifications).
Ca devrait ressembler à ça:



[Corrigé](#)